

GO-09 06: Context

Описание:

В этом задании мы познакомимся с таким понятием, как контекст запроса. Для начала вспомним, что же такое контекст? Контекст - это информация, которая описывает запрос или выполнение какой-нибудь команды.

Контекст запроса

Очень часто контекст используется в HTTP запросе для того, чтобы:

- Передать какое-нибудь значение `context.WithValue`, которое в дальнейшем может быть как-либо использовано. Например, в `middleware` мы можем присвоить запросу идентификатор и положить его в контекст, а затем передать его дальше в заголовке какому-нибудь другому сервису, тем самым делая некое подобие `http` трейсинга (когда по вашему идентификатору можно проследить полный путь клиента через все ваши сервисы).

```
func RequestID(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r
*http.Request) {
        requestID := generateID()
        ctx := context.WithValue(ctx.Background(), "REQUEST_ID",
requestID)
        next.ServeHTTP(w, r.WithContext(ctx))
    })
}
```

- Отмена выполнения запроса `context.WithTimeout` и `context.WithDeadline`, например, когда пользователь неожиданно закрыл страницу, тем самым разорвав TCP соединение, тогда какие-либо действия (запросы к БД, запросы к другим сервисам, логика приложения и т.д) дальше можно уже не делать и не расходовать зря ресурсы системы или можно через `middleware` задать тайм-аут на все запросы к вашему сервису.

```
func RequestTimeout(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r
*http.Request) {
        ctx, cancel := context.WithTimeout(ctx.Background(), 10 *
time.Second)
        defer cancel()
        next.ServeHTTP(w, r.WithContext(ctx))
    })
}
```

Начало http-контекста будет положено при инициализации TCP-соединения, а конец - при его разрыве. Таким образом, у каждого отдельного http запроса есть собственный контекст выполнения, уникальный для этого запроса.

Полезные ссылки:

- [Context в golang приложениях](#)
- [Пакет context go](#)
- [Перевод статьи Go Concurrency Patterns: Context](#)
- [Context canceling](#)
- [log GoDoc](#)
- [os GoDoc](#)

Задание:

В этом задании мы предлагаем вам воспользоваться контекстом, для того чтобы передавать внутри него идентификатор запроса.

Условия:

- В middlewares вашего роутера нужно завести новый middleware RequestIDMiddleware, где положить requestID в контекст. В качестве requestID для примера можете использовать os.Getuid().
- В обработчике CreateUserHandler залогируйте создание пользователя (должно логироваться сообщение, например, user created, а также параметры, с которыми он был создан name, age, email).
- Залогируйте при помощи стандартного логера из пакета log.

Порядок действий:

1. В вашем проекте module09 сделайте новую ветку 06_task.
2. В пакете module09/internal/routers/chi или module09/internal/routers/gorilla создайте RequestIDMiddleware и подключите middleware к роутеру.
3. Проверьте работоспособность.
4. В ответе пришлите ссылку на merge request в ветку master своего проекта ветки 06_task.